DOMINANT TAKEOVER REGIMES FOR GENETIC ALGORITHMS

David Noever
Biophysics Branch, ES76
National Aeronautics and Space Administration
George C. Marshall Space Flight Center
Huntsville, AL 35812 USA

Subbiah Baskaran
Institut fuer Molekulare Biotechnologie, e.V.
Beutenbergerstr. 11, DO-7745, Jena, Germany
Biophysics Branch, ES76, National Aeronautics and Space Administration, George C. Marshall Space Flight Center
Huntsville, AL 35812 USA

The genetic algorithm (GA) is a machine-based optimization routine which connects evolutionary learning to natural genetic laws [1–2]. The present work addresses the problem of obtaining the dominant takeover regimes in the GA dynamics. Estimated GA run times are computed for slow and fast convergence in the limits of high and low fitness ratios. Using Euler's device for obtaining partial sums in closed forms, the result relaxes the previously held requirements for long time limits. Analytical solutions reveal that appropriately accelerated regimes can mark the ascendancy of the most fit solution. In virtually all cases, the weak (logarithmic) dependence of convergence time on problem size demonstrates the potential for the GA to solve large N–P complete problems.

A central issue in how the GA processes strings (solution encoded genomes) is takeover times for the most fit genome [2]. Takeover times here refers to how the computational time or complexity scales with larger problem sizes. Short takeover times correspond to rapid convergence onto the most fit individual. Conversely, long takeover times correspond to slow convergence and sluggish dynamics. A historical discussion of computational complexity appears in reference 3.

Here we solve the standard GA models for generational reproduction [4], evaluate the takeover times, and compare the results using various approximation techniques. The approximations take place only in time and fitness space, such that the exact results can be compared directly. The aim is to identify and understand why the GA sorts through some regions of the fitness landscape so efficiently but gets bogged down in other regions.

For discrete time steps between generations, $t=\{0,1,2...\}$, let $P_{i,t}$ correspond to the proportion of alleles (or bit string values) set to the value 1 for a particular allele position i at generation t. Let $P_{i,o}$ represent P for the founder generation, t=0. For simplicity, all subsequent work will treat binary genetic algorithms which have alleles possessing either of two values, 0 or 1; the multivalue case is a trivial generalization at this level of proof. Let f_1 correspond to the organism's fitness (some survival probability) sampled with allele value 1 in a particular position j. Likewise, take f_o to represent the fitness of all organisms sampled with allele value 0 in position j. For any defined fitness ratio, $r=f_1/f_o$, the value will be considered time-independent and constant across generations.

Thus to begin, assume generational reproduction on a binary fitness landscape (f_o,f_1) with fitness ratio, $r=f_1/f_o$. Previous results have established an equivalence condition to match generational and steady-state reproduction, so we consider the following derivations to develop with some parallel applicability for steady-state GAs (e.g. Genitor, etc.). In the binary GA [5], generational reproduction implies that

$$P_{t+1} = (f_1 / S_t)P_t \qquad (1)$$

where S_t defines the average population fitness, $S_t=f_1P_t+(1-P_t)f_o$.

An iterated recursion relation is complicated by the appearance of P_t in the average population fitness, S_t. Here $S_t=P_tf_1+(1-P_t)f_o$ is the total average population fitness. Physically the last term represents the copying of one individual with a reproductive rate, f_1/S_t.

The recursion relation governing inverse population growth is (X=1/P)[5]:

$$X_t = 1 - \frac{1}{r} + \frac{X_{t-1}}{r} = A + BX_{t-1} \quad . \qquad (2)$$

Iterating and solving as a function of the fitness ratio and time gives:

$$X_t = A\sum_{n=0}^{t} B^n + B^t X_o = \frac{(r-1)}{r}\sum_{n=0}^{t}\left(\frac{1}{r}\right)^n + \frac{X_o}{r^t} \quad . \qquad (3)$$

Equation (3) can be inverted to solve for population dynamics, P_t:

$$P_t = \frac{P_o r^t}{1 + P_o \Gamma_t r^{t-1}(r-1)} \qquad (4)$$

where the term Γ_t is the binomial series

$$\Gamma_t = \sum_{n=0}^{t}\left(\frac{1}{r}\right)^n \quad . \qquad (5)$$

The aim of this analysis is to solve for closed form results in the limits of fitness space for long and short times. The approximate limits will parameterize the GA convergence for arbitrary fitness values.

The Case of Long Times (t-->∞) and Fitness Ratios r>1

For an infinite series summed with large fitness ratios, then for (1/r)<1, the series (5) converges according to

$$\sum_{n=0}^{\infty} x^n = 1 + x + x^2 + ... = (1-x)^{-1} \qquad (6)$$

or

$$\Gamma_t = (1-\frac{1}{r})^{-1} = \frac{r}{r-1} \quad . \qquad (7)$$

Thus, in the limit of long times and fitness ratios greater than unity, the population changes according to the dynamical equation:

$$P_t = \frac{P_o r^t}{1 + P_o r^t} \quad . \qquad (8)$$

Equation (8) can be solved for how the computational complexity varies with the population size, n. Two cases are examined, the worst and average complexities which in turn depend on the expected frequency of the final solution appearing in the initial population, P_o. In the average case, the initial population has a random frequency for the best solution, thus P_o=0.5. Alternatively in the worst case, the initial population has a minimum frequency for the best solution, thus P_o=1/n. Near convergence the final population approaches unity for all cases, thus, P_f=1–(1/n)=(n–1)/n. With this dependence on population size, it is possible to solve directly for the takeover time (or computational complexity) of the most fit member

$$t_c = \frac{\ln\left[P_f / P_o(1-P_f)\right]}{\ln(r)} \quad . \qquad (9)$$

For worst case convergence times, then

$$t_c = \frac{\ln[n(n-1)]}{\ln(r)} \sim O[\ln(n)] \ . \tag{10}$$

Similarly for average convergence times, then

$$t_c = \frac{\ln[2(n-1)]}{\ln(r)} \sim O[\ln(n)] \ . \tag{11}$$

Both cases for takeover times (10–11) demonstrate the same weak logarithmic dependence on population size.

The Case of Both Arbitrary Times and Fitness Ratios

For a finite series summed with arbitrary fitness ratios, then the series (5) converges according to

$$\sum_{n=0}^{t} x^n = 1 + x + x^2 + ... = \frac{(1-x^t)}{(1-x)} \tag{12}$$

or

$$\Gamma_t = \frac{1-(1/r)^t}{1-(1/r)} = \frac{1}{r^{t-1}}\left[\frac{r^t-1}{r-1}\right] \ . \tag{13}$$

Thus, for finite times and arbitrary fitness ratios, the population changes according to the dynamical equation:

$$P_t = \frac{P_o r^t}{1 + P_o r^{t-1}(r-1)\Gamma_t} = \frac{P_o r^t}{(1-P_o) + P_o r^t} \ . \tag{14}$$

This agrees with the result derived independently by Deb and Goldberg [6] and Ankenbrandt [4].

Equation (5) can be solved for the computational complexity with the population size, n and fitness ratio, r.

$$t_c = \frac{\ln\left[\frac{P_f(1-P_o)}{P_o(1-P_f)}\right]}{\ln(r)} \ . \tag{15}$$

Again, two cases are examined, the worst and average complexities. For worst case convergence times, then

$$t_c = \frac{2\ln(n-1)}{\ln(r)} \sim O[\ln(n)] \ . \tag{16}$$

Similarly for average convergence times, then

$$t_c = \frac{\ln(n-1)}{\ln(r)} \sim O[\ln(n)] \ . \tag{17}$$

Both cases for takeover times demonstrate the same weak logarithmic dependence on population size.

The Case of Long Times (t-->∞) and Low Fitness Ratios (r<1)

For an infinite series summed with low fitness ratios, then the series (10) converges according to

$$\Gamma_t = \frac{1-(1/r)^t}{1-(1/r)} = \frac{1}{r^{t-1}}\left[\frac{r^t-1}{r-1}\right] = \lim_{t\to\infty} \frac{r^t-1}{r-1}$$
$$= (1-r)^{-1} \text{ for } r<1 \ . \tag{18}$$

Thus, for infinite times and low fitness ratios, the population changes according to the dynamical equation:

$$P_t = \frac{P_o r^t}{1 + P_o r^{t-1}(r-1)\Gamma_t} = \frac{P_o r^t}{1-P_o} \ . \tag{19}$$

Equation (5) can be solved for the computational complexity with the population size, n.

$$t_c = \frac{\ln\left[P_f(1-P_o)/P_o\right]}{\ln(r)} \ . \tag{20}$$

Again, two cases are examined, the worst and average complexities. For worst case convergence times, then

$$t_c = \frac{\ln\left[(n-1)^2/n\right]}{\ln(r)} \ . \tag{21}$$

For large populations, n-->∞, then the logarithmic numerator goes to the limit, $\ln[n-2+(1/n)] = \ln(n-2)$. For small populations, n-->0, then the logarithmic numerator goes to the limit, $\ln\left[n-2+(1/n)\right] = \ln[(1/n)-2]$.

Similarly for average convergence times, then

$$t_c = \frac{\ln\left[(n-1)/n\right]}{\ln(r)} \ . \tag{22}$$

For large populations, n-->∞, then the logarithmic numerator goes to the limit, $\ln[1+(1/n)] = 0$.

The Case of Finite Times and Arbitrary Fitness Ratios r

For a finite series summed with arbitrary fitness ratios, then the series (5) converges according to an ingenious tool sometimes called Euler's device [7]. For this case, the series is rewritten for $z=(-1/r)$ and the geometric series can be summed for any intermediate time (in powers of $p=2^q$ q=1,2,3...)

$$\sum_{n=0}^{t} z^n = 1 - z + z^2 - z^3 + \ldots \tag{23}$$

$$\sum_{n=0}^{t} z^n = 1 - z + z^2 - z^3 + \ldots \ . \tag{24}$$

To evaluate any intermediate sum, Γ_{2p-1}, Euler's device gives:

$$\Gamma_1 = (1-z) = 1 + 1/r; \Gamma_{2p-1} = \left(1+z^p\right)\Gamma_{p-1} = \left[1+(1/r)^p\right]\Gamma_{p-1} \ . \tag{25}$$

This case represents the main exposition of this paper. It enables one to inspect the GA dynamics in intermediate time intervals without a lack of analytical generality. For any intermediate generation, the series can be summed in terms of previously known terms. As an example, consider the series up to generation 15:

$$\Gamma_3 = \left(1+1/r^2\right)\Gamma_1; \Gamma_7 = \left(1+1/r^4\right)\Gamma_3$$
$$= 1 + 1/r + 1/r^2 + \ldots + 1/r^7; \Gamma_{15} = \left(1+1/r^8\right)\Gamma_7 \ . \tag{26}$$

Thus, in the limit of any finite time and arbitrary fitness ratios, the population changes according to the dynamical equation:

$$P_{2p-1} = \frac{P_o r^{2p-1}}{1 + P_o\left(1+1/r^p\right)\Gamma_{p-1} r^{2p-2}(r-1)} \ . \tag{27}$$

Fig. 1 shows the stepped amplification for discrete iterations using Euler's device on the series sum and the population.

The principal value of the partial summation formalism is to allow intermediate stages of the growth cycle to be expressed in closed form without resorting to any assumptions of long times or constraints on the fitness ratio. The appeal is that programmed changes in population (injection or withdrawal of strings) can be undertaken without loss of analytic versatility. Additionally, the time complexity for regions of rapid convergence (following such injection or withdrawal) can be monitored without resetting the population balance with a new fitness ratio. Finally, the appeal of a 2^q representation space for population changes automatically suggests a simple mapping onto the hypercube of available search space. The generation steps thus naturally can be fitted to vertices of an ever enlarging (hypercube) search space.

Not only does (27) give the partial summation in a closed (polynomial) form, but also allows for the immediate production of interval summations, e.g. the growth that occurred between generations 7 and 15 (or $p-1$ to $2p-1$). Rewriting Euler's device in terms of q for powers of $p=2^q$ gives:

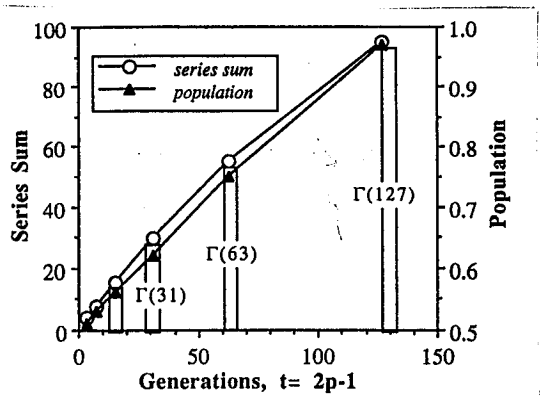

Fig. 1. Stepped population growth and series sum using Euler's device to obtain partial summations without constraining the fitness space or simulation time. The bars highlight the points of partial summation on a binary landscape (t=$2p$−1 where p=2^q).

$$q:\Gamma_{2^{q+1}-1}=\left(1+z^{2^q}\right)\Gamma_{2^q-1} \tag{28}$$

$$q-1:\Gamma_{2^{q-1}-1}=\left(1+z^{2^{q-1}}\right)\Gamma_{2^{q-1}-1}\ . \tag{29}$$

By subtracting (29) from (28) and rewriting in terms of p gives the interval summation formula:

$$\Gamma_{2p-1}-\Gamma_{p-1}=\Gamma_{2p-1}-\left(1+z^{p/2}\right)\Gamma_{p/2-1}\ . \tag{30}$$

For example, the population growth between generations 7 and 15 can be written in exact closed form as:

$$\Gamma_{15}-\Gamma_7=\Gamma_{15}-\left(1+1/r^4\right)\Gamma_3\ . \tag{31}$$

As shown schematically in Fig. 2, a complete interval analysis becomes possible for subsets of the partial summation. By breaking the population growth curve into discrete intervals, a useful formalism evolves for addressing intermediate changes in GA dynamics (e.g. withdrawal or addition of strings, changes in fitness). Thus the objective of tracking regions of accelerated convergence is simplified without loss of mathematical generality.

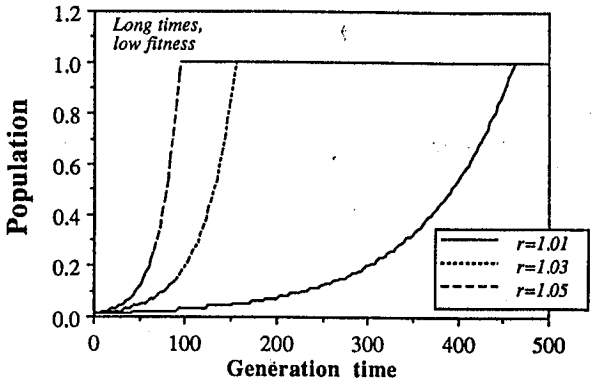

Fig. 2. Schematic of interval summations within the partial sums of Euler's device.

The Case of Arbitrary Times (t-->∞) and Fitness Ratios r=1

For completeness sake, consider the trivial case of equal fitness, r=1. When the two solutions have indistinguishable performance, the population should remain fixed on average at the initial value, P_o. As a check on the previous formalism, such a result does indeed follow when the infinite series sums for equal fitness, then for (1/r)=1 the series (5) converges according to

$$\sum_{n=0}^{t}x^n=1+x+x^2+\ldots=t \tag{32}$$

or

$$\Gamma_t=1+1+1+\ldots+1=t\ . \tag{33}$$

Thus, in the limit of arbitrary times and equal fitness ratios, the population remains unchanged according to the dynamical equation:

$$P_t=\frac{P_o r^t}{1+P_o t\, r^{t-1}(r-1)}=P_o\ . \tag{34}$$

A somewhat shorter derivation follows from observing that (r−1)=0 in this case, thus for any finite number of terms in the summation, the dependence on time should vanish entirely from the denominator of (30).

Summary of Results

A summary of the various approximations is shown in Fig. 3. The graphical difference between exact and approximate population dynamics is highlighted in Fig. 4. The time complexities calculated in the various approximate limits are compared in Fig. 5. In general the weak (logarithmic) dependence of the GA processing on problem size gives it great potential compared to other sorting and search routines (Fig. 6).

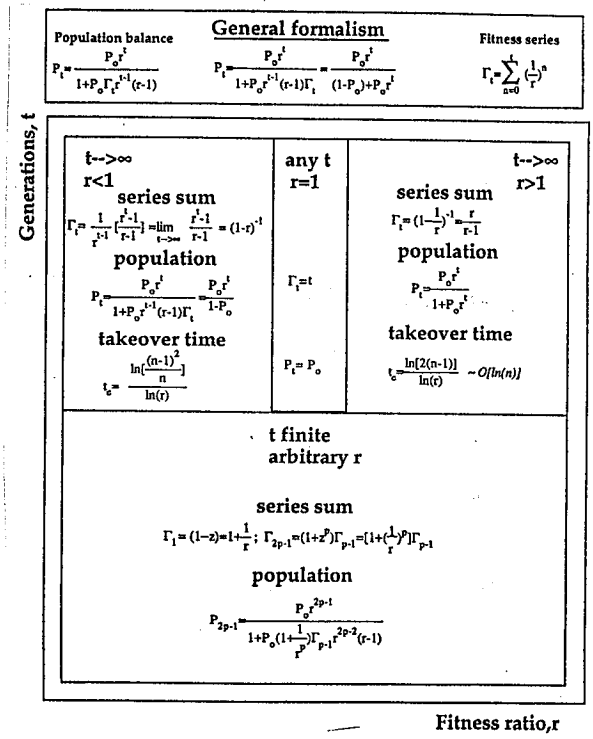

Fig. 3. Phase diagram for series summation in approximate limits of short and long times and arbitrary fitness values.

Much work has been published on the compromises struck between rapid GA convergence and processing parameters (e.g. reference 6). A preliminary discussion here presents the fundamental issue in terms of accelerated regions of GA convergence. In general, GA populations move toward optimum following an S-curve or logistic growth. The central steep ascendency of the S-curve corresponds to highly efficient GA processing. In this regime, genetic mixing improves overall performance, such that neither good strings get heavily disrupted (strong crossover) nor do bad strings get unnecessarily preserved (low selection). Thus GA efficiency can match with the genetic terms of a balanced tradeoff struck between diversity

and selectivity. Over time, this tradeoff can be imagined schematically to combine two composite operators. For the same operations mapped into time and fitness space, neighborhoods of high diversity correspond to high fitness ratios, while neighborhoods of strong selectivity correspond to high fitness ratios. In this translation, regimes of accelerated convergence can be identified directly with their approximate time complexities (Fig. 3).

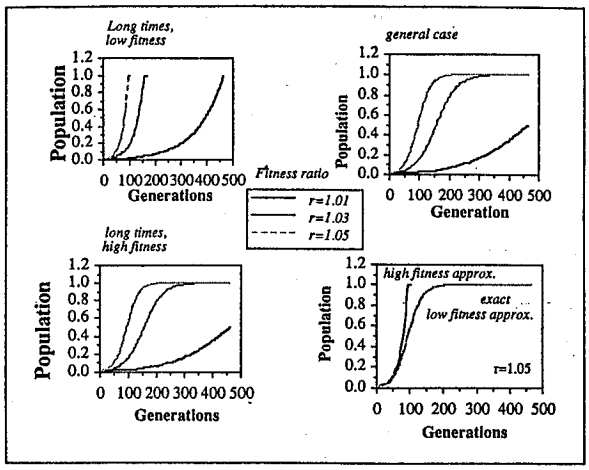

Fig. 4. Summary of low and high fitness approximations and their effects on population growth. Lower right shows the comparison between exact and approximate solutions. Note that the exact and high fitness regimes essentially overlap.

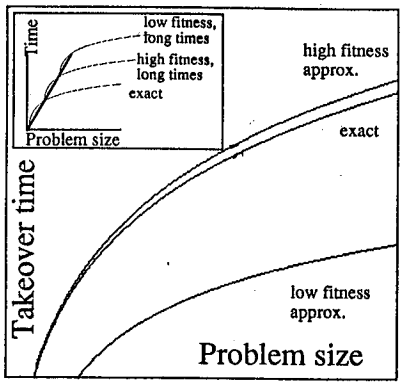

Fig. 5. Comparative time complexities (takeover times) for the approximate limits. Short times correspond to regions of accelerated convergence.

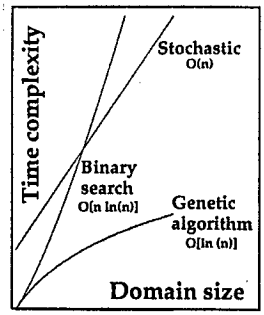

Fig. 6. Comparative computational complexity between various sort routines.

To summarize, the work has analyzed the population dynamics of a general GA in fitness space. The iterated population yields recursion relations and a convenient formalism is developed for the approximate cases of long and short times as well as high and low fitness values. The validity of these closed forms is compared to the exact result. By using the elegant tool of partial summation (Euler's device) then the analytical basis of the GA is prepared to handle discontinuous changes in parameters without loss of mathematical generality. For example, the introduction or removal of strings can be safely accounted for by using the discrete summation formalism in a particularly transparent way. An additional advantage in using the partial series summation (Fig. 1) is to relax the requirements for long (infinite) time series. In practice, most GAs run no longer than a few decades of generational time steps. Future work will examine the potential for abrupt population changes or more dynamic fitness landscapes (time dependent optimizations) in a focused effort to understanding GA processing and convergence.

References

1. Holland, J.H.: Adaptation in Natural and Artificial Systems. Ann Arbor, Michigan: University of Michigan Press 1975.
2. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Reading, Massachusetts: Addison Wesley 1989.
3. Cook, S.A.: Comm. of ACM 26, 401 (1983).
4. Ankenbrandt, C.: (1991) "Time Complexity and Convergence of Genetic Algorithms".Rawlins, G.J.E. (ed.): Foundations of Genetic Algorithms. San Mateo, California: Morgan Kaufmann Publishers 1991.
5. Noever D. and Baskaran, S.: "Steady State vs. Generational Genetic Algorithms: A Comparison of Time Complexity and Convergence Properties" Santa Fe Institute preprint series, 92-07-032 (submitted to Machine Learning) (1992).
6. Goldberg, D.E. and Deb, K.: "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms," Rawlins, G.J.E. (ed.): Foundations of Genetic Algorithms. San Mateo, California: Morgan Kaufmann Publishers 1991.
7. Broucke, R.A.: Comm. of the ACM 14, 34 (1971).